



SMTP Deux

Developer Documentation v1.0.0

©1998-2001 Deep Sky Technologies, Inc. All Rights Reserved.
Published and Distributed Worldwide by Deep Sky Technologies, Inc.

Deep Sky Technologies, Inc.
P.O. Box 6897
Vero Beach, FL 32961-6897
(561) 794-9494



<http://www.deepskytech.com/>

Software Engineers

James T. Crate
Robert T. McGoye
Steven G. Willis

Manual

Steven G. Willis

Software License and Limited Warranty

Please read this license carefully before using the software. By using the software, you agree to become bound by the terms of this agreement, which includes the software license and warranty disclaimer (collectively referred to herein as the "agreement"). This agreement constitutes the complete agreement between you and Deep Sky Technologies, Inc. If you do not agree to the terms of this agreement, do not use the software and promptly destroy all copies in your possession, physical and electronically.

1. Ownership of Software: The enclosed manual and computer programs ("Software") were developed and are copyrighted by Deep Sky Technologies, Inc. ("DSTi") and are licensed, not sold, to you by DSTi for use under the following terms, and DSTi reserves any rights not expressly granted to you. DSTi retains ownership of all copies of the Software itself. Neither the manual nor the Software may be copied in whole or in part except as explicitly stated below.

2. License: DSTi, as Licensor, grants to you, the Licensee, a non-exclusive, non-transferable right to use this Software subject to the terms of the license as described below:

- a. You may make backup copies of the Software for your use provided they bear the DSTi copyright notice.
- b. You may use this Software in an unlimited number of distributed copies of a single application or database. Use in additional applications requires separate licenses for the use of this Software.
- c. Distribution or dissemination of the software license serial is strictly prohibited. This license grants the original licensee sole use of the license serial for enabling this Software.

3. Restrictions: You may not distribute copies of the Software to others (except as an integral part of a database or application within the terms of this License) or electronically transfer the Software from one computer to another over a network. You may distribute copies of the Software as an integral part of a development shell or non-compiled commercial database as long as

the DSTi copyright notices and documentation remain intact with the distribution. The Software contains trade secrets and to protect them you may not decompile, reverse engineer, disassemble, or otherwise reduce the Software to a human perceivable form. You may not modify, adapt, translate, rent, lease, loan or resell for profit the software or any part thereof.

4. Termination: This license is effective until terminated. This license will terminate immediately without notice from DSTi if you fail to comply with any of its provisions. Upon termination you must destroy the Software and all copies thereof, and you may terminate this license at any time by doing so.

5. Update Policy: DSTi may create, from time to time, updated versions of the Software. At its option, DSTi will make such updates available to the Licensee.

6. Warranty Disclaimer: The software is provided "as is" without warranty of any kind, either express or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. DSTi does not warrant, guarantee, or make any representations regarding the use, or the results of the use, of the software or written materials in the terms of correctness, accuracy, reliability, currentness or otherwise. The entire risk as to the results and performance of the software is assumed by the Licensee. If the software or written materials are defective you, and not DSTi or its dealers, distributors, agents, or employees, assume the entire cost of all necessary servicing, repair or correction. No oral or written information or advice given by DSTi, its dealers, distributors, agents, or employees shall create a warranty or in any way increase the scope of this warranty, and you may not rely on such information or advice. This warranty gives you specific legal rights. You may have other rights, which vary from state to state.

7. Governing Law: This agreement shall be governed by the laws of the State of Florida.

Copyrights and Trademarks

All trade names referenced in this document are the trademark or registered trademark of their respective holder.

BASh, TCP Deux, SMTP Deux, POP3 Deux, FTP Client Deux, HTTP Client Deux, eTrans, TCP Server Deux, and HTTP Server Deux are copyright Deep Sky Technologies, Inc.

4th Dimension, ACI, ACI US, 4D Compiler, 4D, 4D Server, 4D Client, and 4D Insider are trademarks of 4D, Inc.

4D Internet Commands plugin provided courtesy, and with permission, of 4D, Inc.

Internet Commands plugin provided courtesy, and with permission, of Christian Quest.

Macintosh and MacOS are trademarks of Apple Computer, Inc.

Windows is a trademark of Microsoft Corporation.

Table of Contents

Items in **Bold** were just added in the latest release of the SMTP Deux component.
Items in *Italic* have not had their content filled in completely as of yet.
Items in Underline have had important updates since the last release of the SMTP Deux component.

Software License and Limited Warranty

Copyrights and Trademarks

Table of Contents

Preface

Acknowledgements

Features

 System Requirements

 Support

Components

 Installing and Updating SMTP Deux

 Managing Installation Conflicts

 Affix SMTP Deux Document

 Uninstalling SMTP Deux

Basics of SMTP

Mail Header Lines

 Formatted Recipients

Recipients Versus Header Lines

 Formatted Attachments

Content Types

Constants

 SMTPd_Header_Codes

Methods

ENV_Get_SMTPd_HardName_Long

ENV_Get_SMTPd_HardName_Short

 INIT_SMTPd

 SMTPd_ERROR

 SMTPd_Extract_EmailAddress

 SMTPd_Extract_FullName

 SMTPd_Get_Attachment_w_Header

 SMTPd_Get_Recipient_Formatted

 SMTPd_Send_Message_Attachment

 SMTPd_Send_Message_Complex

 SMTPd_Send_Message_Simple

Version History

SMTP Deux v1.0.0

 SMTP Deux v1.0.0b03

SMTP Deux v1.0.0b02
SMTP Deux v1.0.0b01
SMTP Deux Error Codes
Index

Preface

The SMTP Deux component is designed to work in conjunction with many other components. Specifically, the SMTP Deux component requires that the BASH component, available for free from DSTi, and TCP Deux component both be installed already in your database structure file.

Make certain that you view the compatibility matrix for components available to make certain you are using compatible versions of the different components required. There is a compatibility matrix available in this manual; the most recent compatibility matrix is available on the DSTi web site.

Acknowledgements

The creation of the SMTP Deux component is not directly attributable to any single person. Particular pieces of functionality within the SMTP Deux component may be from the direct knowledge and experience of certain developers, but the overall concept and construction of the SMTP Deux component has come from all of the developers at Deep Sky Technologies, Inc.

In particular, the tireless efforts of Robert T. McGoye have contributed the most to the SMTP Deux component. His ability, and patience, to be able to tolerate the swings in the atmosphere at DSTi, have proven to be invaluable in the development of the SMTP Deux component.

Later tweaks and additions to the SMTP Deux component have resulted from the training of James T. Crate. Mr. Crate's experience in many different programming environments has provided refreshing insights into the overall structure and organization of the core routines at DSTi, the same core routines which are available in the BASH and SMTP Deux components.

Finally, I, Steven G. Willis, might have had something to do with the creation of the SMTP Deux component...

Features

SMTP Deux is a 4th Dimension component that provides a set of SMTP routines for sending email from 4D. SMTP Deux works on top of TCP Deux, providing a common code set for sending email that will work with all of the major TCP plugins available currently for 4D (4D Internet command v6.7.x, Internet ToolKit v2.0.x, and Internet ToolKit v2.5.x).

SMTP Deux is made available for free. Of course, it does require the purchase of TCP Deux to work properly. But, with TCP Deux, SMTP Deux provides then full email sending functionality without any of plugin specific drawbacks of any single TCP plugin.

The single greatest feature of SMTP Deux is that it does not use the SMTP layer within 4D Internet Commands. Over the years, many developers have encountered different bugs and anomalies within the SMTP implementation of 4D Internet Commands. SMTP Deux uses the routines within the TCP Deux component package for all TCP communications, which themselves only use the TCP layer within 4D Internet Commands when that plugin is in use. So, most if not all errors which 4D developers experience when sending email with 4D Internet Commands are removed when using SMTP Deux.

As well, SMTP Deux provides a full and complete implementation of the SMTP protocol. So, for those developers using either version of Internet ToolKit and experiencing difficulty setting up their own email sending routines, there is no longer any need to go through this hassle. SMTP Deux provides you with successive email sending routines so that you can choose the implementation that works best for your email sending needs.

SMTP Deux includes full compatibility with SMTP AUTH, authenticated login for SMTP servers. When enabled, SMTP Deux will choose the most secure means supported by the SMTP server to login. This includes CRAM-MD5, Login, and Plain SMTP AUTH mechanisms. There is currently no other solution available in 4D that provides complete support for SMTP AUTH.

All of the code within the SMTP Deux component is ready to use immediately. There are no compiler issues to be concerned about; all of the variables are typed in compiler methods for even the most

stringent 4D developers. Just install the component in your structure and start calling the methods within it. It is really that simple!

System Requirements

The SMTP Deux component is compatible with both Macintosh and Windows installations of 4th Dimension.

Since it is a component, it does require at least version 6.7 of 4th Dimension or above, including 4D Insider v6.7 or above for installation.

Other than the normal hardware and software requirements for your version of 4th Dimension, there are no other minimum requirements for proper use of this software.

Support

Support is provided for SMTP Deux component free of charge for all currently licensed users. Included support services provided for all currently licensed users encompasses all of the online support services available through the DSTi web site (email, FAQ, messaging, etc.). Check the DSTi web site for current direct support options available; we are always working to offer more resources for your needs.

Contact information, including email address(es), phone number(s), and a Contact Us request form, for Deep Sky Technologies, Inc., can be found on the DSTi web site located at:

<http://www.deepskytech.com/>

If there are terms or conventions which you find difficult to understand in relation to the SMTP Deux component or TCP networks in general, feel free to contact Deep Sky Technologies, Inc., support. We will be more than happy to help you in any way we reasonably can. And, only through your questions do we know what subjects to include in future versions of this manual.

Components

A component groups various 4D objects (tables, project methods, forms, menu bars, variables, etc.) representing one or more additional functions. Developing a 4D component providing electronic mail functionality is one such example. A component is autonomous and must be able to be installed in any 4D structure.

Components are defined, generated, and installed with the help of 4D Insider. The component definition is based on the cross referencing analysis performed by 4D Insider (target objects and source objects).

Unlike libraries and groups, components embed the idea of security of objects that they compose. During the development phase of the component, each object is attributed an access type, "Public", "Protected" or "Private". This attribute determines whether each object will be visible or modifiable in 4th Dimension and in 4D Insider once the component is installed within a 4D database.

Installing and Updating SMTP Deux

Installing SMTP Deux or updating an existing version of SMTP Deux within a 4D database is performed using 4D Insider. The activity primarily consists of installing the SMTP Deux component in a database structure opened with 4D Insider (installing the TCP Deux component in a library is not supported at this time).

4D Insider will manage possible conflict issues within the installation and will inform you as they are detected. Though, with the naming conventions used within the SMTP Deux component and the limited number of object names, conflicts should be very rare.

To install or update the SMTP Deux component, follow these very simple steps:

Open the uncompiled structure that you wish to install SMTP Deux into using 4D Insider.

Choose the "Install/Update..." command in the "Components" menu.

A standard open file dialog box will appear.

Select the SMTP Deux component file and click on the "Open" button.

4D Insider parses the SMTP Deux component and prepares to integrate it with your open database. 4D Insider will detect if the operation is an installation or an update of the SMTP Deux component.

In the event of a new installation, all SMTP Deux objects are installed.

In the event of an update, 4D Insider compares the version numbers of both the currently installing SMTP Deux component and the already installed SMTP Deux component. If the date of the "new" component is older than the already installed component, a dialog box will

alert you, allowing you to then "Continue" or "Cancel" the update.

4D Insider replaces old objects with newer objects within the SMTP Deux component and adds new objects from the new SMTP Deux component. 4D Insider takes into account "public" objects having been modified by you (e.g. "_ERROR" methods) and will prompt you to either save or replace them. If any other conflicts arise from the installation or update of the SMTP Deux component, 4D Insider will prompt you with an appropriate dialog box.

Save the database in 4D Insider.

Place a copy of the Affix SMTP Deux document in the 4DX folder.

The Affix SMTP Deux document (entitled **Affix_SMTp_Deux** on Macintosh) contains essential data, resources and constants for many of the methods within the SMTP Deux component. For many of the methods within the SMTP Deux component to function properly, the Affix SMTP Deux document must be in the 4DX folder for the current structure.

On Macintosh, the Affix SMTP Deux document is entitled **Affix_SMTp_Deux** and is located in the Mac4DX directory in the SMTP Deux component's archive. The document should be copied into the Mac4DX folder of your current structure. If the SMTP Deux component is going to be used in all of your 4D projects, the Affix SMTP Deux document can instead be placed within the Mac4DX folder within the 4D folder of your system.

On Windows, the Affix SMTP Deux document is actually two documents: **Afx_SMTpD.4DX** and **Afx_SMTpD.RSR**. These two documents correspond to the data fork and the resource fork of the Affix SMTP Deux document used on Macintosh. These documents are located in the Win4DX directory in the SMTP Deux component's archive. These documents should be copied into the Win4DX folder of your current structure. If the SMTP Deux component is

going to be used in all of your 4D projects, the Affix SMTP Deux document can instead be placed within the Win4DX folder within the 4D folder of your system.

For client/server installations in cross-platform environments, both the Macintosh and Windows versions of the Affix SMTP Deux document should be installed.

Call the method *INIT_SMTPd* early in the On Startup database method.

To initialize the SMTP Deux component in your code, place a call to the method *INIT_SMTPd* early in your **On Startup** database method.

Details about the *INIT_SMTPd* method can be found in the method documentation section of this manual, below.

The SMTP Deux component is now installed/updated in your database and is listed on the "Components" page of the 4D Explorer.

Managing Installation Conflicts

On very rare occasions, when the SMTP Deux component is installed or updated in your 4D database, several questions and conflicts may arise. In the event of an update, 4D Insider will detect that you have modified one of more "Public" objects in SMTP Deux after the initial installation. Or, one or more objects of the same type and of the same name may already exist in your database and in the SMTP Deux component.

4D Insider detects and solves these conflicts during installation:

Modified public objects (updates only)

In this case, 4D Insider alerts you by a dialog box, allowing you to choose an update mode:

Replace the object

Replace all objects

Do not replace the object

Stop installation

Name conflicts

In this case, 4D Insider stops the SMTP Deux installation process, alerts you through a dialog box and saves the list of objects in conflict. This list is stored as a text file in the 4D database folder.

Naming conflicts between logical objects, such as variables, are managed by 4D Insider, in a manner that allows database compilation and avoids conflicts between SMTP Deux and other 4D components.

It may be necessary to rename certain objects in your database or in other components in order to be able to install the SMTP Deux .

If any naming conflicts do occur between SMTP Deux and other 4D components, please notify Deep Sky Technologies, Inc., immediately.

Affix SMTP Deux Document

The Affix SMTP Deux document (entitled **Affix SMTP Deux** on Macintosh) contains essential data and resources for many of the methods within the SMTP Deux component. For many of the methods within the SMTP Deux component to function properly, the Affix SMTP Deux document must be in the 4DX folder for the current structure. For distributed and installed versions of your 4D projects, the Affix SMTP Deux document must be available in the 4DX folder for the SMTP Deux methods to continue to function properly.

Note: it is best to consider the Affix SMTP Deux document another plug-in within your 4D project. Though there no actual plug-in calls available within the Affix SMTP Deux document, it does contain data and resources essential to the operation of the SMTP Deux methods. The SMTP Deux component has been designed to find the Affix SMTP Deux document correctly in all environments (any platform, any 4DX folder, single user or clients/server, etc.). Since the Affix SMTP Deux document is configured similarly to plug-ins, 4D and the SMTP Deux component will automatically manage the document for you in all of the possible installations of a 4D project.

Uninstalling SMTP Deux

4D Insider allows you to uninstall the SMTP Deux component from your 4D database.

To uninstall SMTP Deux from your 4D database:

Using 4D Insider, open your database containing the copy of SMTP Deux to be uninstalled.

In the "Main" listing window, select the SMTP Deux component.

Consider again how great the SMTP Deux component is and make certain that you will *really* no longer need it in your 4D database.

Select the "Uninstall..." command in the "Components" menu.

This command is only active when a component is installed in the database. A dialog box appears allowing you to confirm or cancel the operation. If you uncertain about the previous step then the cancel option is probably your best choice at this time.

Click "OK" to validate the operation.

Remove the Affix SMTP Deux document and internet connectivity plugins from your 4DX folder.

Remove the call to the method *INIT_SMTPd* from your On Startup database method.

All objects from the SMTP Deux component are deleted from your 4D database. Obviously, you are now very sad to no longer have the SMTP Deux component in your 4D database. Crying is allowed...

Basics of SMTP

SMTP, the Simple Mail Transfer Protocol, is an Internet standard for sending email. SMTP is used to not only to begin the sending process, that of transferring an email message from a regular email client to an email server, but SMTP is also used to deliver email from an originating email server to a destination email server. In other words, the exact same protocol that is used to start the sending of an email is also used to transfer email from place to place on the Internet.

The procedure for implementing SMTP provides a mechanism for the originator and destination servers and addresses to be specified independent of the actual message contents. It is important to understand that this mechanism for specifying the originator and destination data is independent of the actual email content; and, the email content includes not just the message and subject being sent, but the headers lines which included with the email message. Because of this disparity, it is a simple matter for the source and destination information to be reasonably masked from the message which is actually delivered, hence the complexity of preventing spam (unsolicited email messages).

This section tries to provide the basic information needed to understand the different features of any SMTP system. In no way is this explanation complete. But, it should be enough to provide most 4D implementations of SMTP with more than enough details to do whatever is reasonably needed.

For more information regarding the SMTP protocol, it is recommended that the reader refer to RFCs 821 (SMTP), 822 (SMTP Headers), 2554 (SMTP AUTH), and 2822 (Text Message Formats). You can download any of these RFCs by following these links:

http://www.deepskytech.com/rfc/rfc_0821.txt

http://www.deepskytech.com/rfcs/rfc_0822.txt

http://www.deepskytech.com/rfcs/rfc_2554.txt

http://www.deepskytech.com/rfcs/rfc_2822.txt

Mail Header Lines

Within the contents of an SMTP message, commonly called an email message, there are standard header lines used as indicators for routing, recipient, sender, priority, content details, and message details. These header lines have a specific format that for most purposes consists of a named value pair delimited by a colon. Values which are exceptionally long can wrap to multiple lines (a new line being defined as a carriage return followed by a linefeed) by subsequent lines beginning with a space; often, these lines are called "folded" lines.

Some mail header lines are added by a mail server when it routes a message. Most email header lines are the responsibility of the originating sender host to include correctly.

Within SMTP Deux, the setting of the actual header lines are handled internally by the methods **SMTPd_Send_Message_Simple** and **SMTPd_Send_Message_Attachment**. Only when greater control over the actual contents of the message being sent should the method **SMTPd_Send_Message_Complex** be needed by the developer. And, when using this method, the developer has almost complete control when using SMTP Deux to set all of the header lines within a message.

SMTP Deux includes a set of constants, **SMTPd_Header_Codes**, for specifying the commonly supported header line types. It is important to format the data for each header line correctly. RFC 822 is the best source of information for the actual format of the different header lines. RFC 822 can be downloaded from:

http://www.deepskytech.com/rfc/rfc_0822.txt

Often, SMTP server will not restrict messages that contain improperly formatted header lines. But, this can

cause problems for the recipients of messages as it may make it difficult if not impossible to read the message.

Formatted Recipients

For header lines that specify recipients, or senders, of an SMTP message, there is a particular format for the values which is worth going over. These header lines include the following:

Constant
SMTPd_FROM_Type
SMTPd_TO_Type
SMTPd_CC_Type
SMTPd_BCC_Type
SMTPd_Reply_To_Type
SMTPd_Sender_Type
SMTPd_Resent_Reply_To_Type
SMTPd_Resent_From_Type
SMTPd_Resent_Sender_Type
SMTPd_Resent_To_Type
SMTPd_Resent_CC_Type
SMTPd_Resent_BCC_Type

For these header lines, a recipient, whether that be an account receiving the message, a sender, or another account referenced, will consists of multiple pieces of information. For all practical purposes, these pieces of information will be the recipients name and recipients email address.

For instance, if a message were to be sent to Steven G. Willis at Deep Sky Technologies, Inc., then the recipient's name would be "Steven G. Willis" and the recipient's email address would be "sgwillis@deepskytech.com".

The formatting for these values in a header line should always be the recipient name in double quotations followed by a space, and then the recipient address in angled brackets. So, a properly formatted recipient for the above example would be:

"Steven G. Willis" <sgwillis@deepskytech.com>

Multiple recipients within a single header line should be delimited by a comma and a space.

There are methods within SMTP Deux to help with the creation and parsing of formatted recipient values. See the section documenting the methods `SMTPd_Get_Recipient_Formatted`, `SMTPd_Extract_EmailAddress`, and `SMTPd_Extract_FullName` for more details on these methods.

Recipients Versus Header Lines

As mentioned previously, the actually originating and destination address of an SMTP message is set separately from the values which are set into the header lines of a message. Spammers often take advantage of this fact to send spam to email accounts; and, it is not uncommon for a recipient of such spam to wonder how they received it when they are not even listed as a recipient in the headers of the message.

For practical reasons, SMTP Deux does **not** discriminate between these two; in other words, even when using the method `SMTPd_Send_Message_Complex`, the recipients of an SMTP message are pulled directly from the headers lines for TO:, CC:, and BCC: when the message is actually send by SMTP Deux. Of course, BCCed addresses are not included within the headers of a message that is being sent, but this does not preclude the need for at least one value TO: recipient to be included within the headers of an SMTP message sent with SMTP Deux.

Attachments

When attachments are sent with an SMTP message, the attachment is actually encoded and formatted in a particular way within the body of the message. The encoding of the attachment assures for the proper transmission of the document within the constraints of transport restrictions; universally, base64 encoding of an attachment is adequate to assure the proper deliver of any attachment type to any recipient program. The formatting of the encoded attachment will included information pertaining to the encoding type used, the original document title, and the type of content of the attachment.

SMTP Deux can send attachments with the methods **SMTPd_Send_Message_Attachment** and **SMTPd_Send_Message_Complex**. But, for each method, the handling of attachments is different.

When using the method **SMTPd_Send_Message_Attachment**, SMTP Deux will handle internally the encoding and formatting of the attachment when sending the message. All that is required is that the document contents, document title, and document content type be passed in as parameters. SMTP Deux will handle the details of properly encoding and formatted the attachment for inclusion with the message.

When using the method **SMTPd_Send_Message_Complex**, it is the responsibility of the developer to properly encode and format the attachment. A utility method, **SMTPd_Get_Attachment_w_Header**, is included with SMTP Deux; this method will take as parameters the contents of the document, the document title, and the document content type and will format the data for passing to the method **SMTPd_Send_Message_Complex**. If the developer wishes to provide more detailed or alternate encoding or formatting for attachments sent using

SMTPd_Send_Message_Complex, then it is the responsibility of the developer to properly encode and format the attachment for inclusion with an SMTP message.

Content Types

When sending an SMTP message, the content type of the message body must be specified. Using the methods `SMTPd_Send_Message_Simple` and `SMTPd_Send_Message_Attachment`, this is handled automatically by SMTP Deux. But, when using the method `SMTPd_Send_Message_Complex`, it is up to the developer to set the content type of the message body.

Common values for content type include:

- text/plain
- text/html
- image/gif
- image/jpeg
- application/pdf
- application/octet-stream
- application/x-macbinary
- application/x-stuffit
- multipart/mixed

Depending on the contents of the message being sent, different content types should be used. Most commonly, for an SMTP message, either text/plain or multipart/mixed are used; the former is for a standard text message and the latter is for a message that contains multiple parts (e.g. attachments).

Recently, text/html is becoming more and more popular, whereby an SMTP message consists solely of "HTML email" in the body of the message. It is recommended though that if the goal is to send HTML email, then the HTML should be sent as an attachment with the HTML message and the raw text of the message be included for those recipients that do not support the viewing of HTML email.

For SMTP messages which have a content type of multipart/mixed, the different "parts" of the message body must be separate by a boundary. This boundary is an indicator for the receiving software to properly delimit the different sections of the message body. The boundary

is usually formatted within the value of the content type header as:

```
multipart/mixed; boundary="bondary_here"
```

An example header line within an SMTP message for a content type with an attachment might look like:

```
Content-Type: multipart/mixed; boundary="-----  
---E44A07500B354C3BB0E42090"
```

For attachments included with an SMTP message, a content type must be provided also in the headers of the attachment. This is used to indicate the content type of the encoding attachments which follows. Obviously, an attachment with a content type of multipart/mixed makes little sense at this point within the header of the attachment.

Constants

There are a minimal number of custom constants included with the SMTP Deux component package. These constants are grouped into a single, convenient constant groups for easier referencing and organization.

Where appropriate, it is highly recommended that the custom constants included with the SMTP Deux component be utilized within your code; this will simplify considerably future feature enhancements to the core code within SMTP Deux.

SMTPd_Header_Codes

The SMTPd_Header_Codes constants group contains one constant for each of the common SMTP header lines tags supported within the SMTP Deux component package. The following is a listing of the constants, their values, and the text of the actual header line in email corresponding to each, within the SMTPd_Header_Codes constant group:

<u>Constant</u>	<u>Value</u>	<u>Text of Header Line</u>
SMTPd_FROM_Type	1	From
SMTPd_TO_Type	2	To
SMTPd_CC_Type	3	CC
SMTPd_BCC_Type	4	BCC
SMTPd_Reply_To_Type	5	Reply-To
SMTPd_Sender_Type	6	Sender
SMTPd_Resent_Reply_To_Type	7	Resent-Reply-To
SMTPd_Resent_From_Type	8	Resent-From
SMTPd_Resent_Sender_Type	9	Resent-Sender
SMTPd_Orig_Date_Type	10	Orig-Date
SMTPd_Date_Type	11	Date
SMTPd_Resent_Date_Type	12	Resent-Date
SMTPd_Resent_To_Type	13	Resent-To
SMTPd_Resent_CC_Type	14	Resent-CC
SMTPd_Resent_BCC_Type	15	Resent-BCC
SMTPd_In_Reply_To_Type	16	In-Reply-To
SMTPd_References_Type	17	References
SMTPd_Keywords_Type	18	Keywords
SMTPd_Subject_Type	19	Subject
SMTPd_Comments_Type	20	Comments
SMTPd_Encrypted_Type	21	Encrypted
SMTPd_Mime_Ver_Type	22	Mime-Version
SMTPd_Content_Type_Type	23	Content-Type
SMTPd_X_Mailer_Type	24	X-Mailer

In general, these constants will probably not be used within your coding with the SMTP Deux component. The only exception to this is if you use the SMTPd method *SMTPd_Send_Message_Complex* in which case you will be using these values directly.

Methods

All of the available methods within the SMTP Deux component are documented here. Each parameter and return value is listed, including parameter types and descriptions. Full descriptions of the functionality of each method is also provided.



ENV_Get_SMTPd_HardName_Long

ENV_Get_SMTPd_HardName_Long => *Long Hard Name*

ENV_Get_SMTPd_HardName_Long
=> *Long Hard Name* :Text

Parameter	Type	Description
<i>Long Hard Name</i>	Text	Full, hard coded name of SMTP Deux component including versioning information

The method **ENV_Get_SMTPd_HardName_Long** returns the full, hard coded name of the SMTP Deux component, including versioning information.

Long Hard Name is the full, hard coded name of the TCP Deux component. As of this release, this will always return the value "SMTP_Deux_v1.0.0".

Note: this method was added as of SMTP Deux v1.0.0.



ENV_Get_SMTPd_HardName_Short

ENV_Get_SMTPd_HardName_Short => *Short Hard Name*

ENV_Get_SMTPd_HardName_Short
=> *Long Hard Name* :Text

Parameter	Type	Description
-----------	------	-------------

Short Hard Name Text

Full, hard coded name of SMTP Deux component including versioning information

The method *ENV_Get_SMTPd_HardName_Short* returns a shortend, hard coded name of the SMTP Deux component.

Short Hard Name is the shortened, hard coded name of the SMTP Deux component. As of this release, this will always return the value "SMTP_Deux".

Note: this method was added as of SMTP Deux v1.0.0.



INIT_SMTPd

INIT_SMTPd

INIT_SMTPd

Parameter	Type	Description
<i>none</i>	none	none

The method *INIT_SMTPd* initialises the SMTP Deux component. A single call to this method should be made early in the On Startup database method in your 4D application. Make certain the call to this method follows the initialization call to the BASH and TCP Deux components but before any other calls to the SMTP Deux component package.



SMTPd_ERROR

SMTPd_ERROR (*SMTPd Error Number ; Special Error Text ; Calling Method Name*)

SMTPd_ERROR

(

-> *SMTPd Error Number*: **Longint**


```

-> Special Error Text: Text
-> Calling Method Name: Text
)

```

Parameter	Type	Description
<i>SMTPd Error Number</i>	Longint	Internal SMTPd error number
<i>Special Error Text</i>	Text	Special text to describe the exact error instance
<i>Calling Method Name</i>	Text	Name of the method that the error condition occurred in

The method ***SMTPd_ERROR*** acts as a callback method from within the SMTP Deux component for errors that may occur. Any time an error condition is detected within the SMTP Deux, a call to the method ***SMTPd_ERROR*** is made.

The internal *SMTPd Error Number* is passed to this method as the first parameter. The *Special Error Text* parameter will contain any relevant error text which is specific to the error which occurred. It is not uncommon for the *Special Error Text* value to be empty. The *Calling Method Name* will always contain the name of the SMTP Deux method which called the ***SMTPd_ERROR*** method.

The ***SMTPd_ERROR*** method has been implemented as a source for a consistent interface and/or error tracking mechanism to be available while using the SMTP Deux component. This method can be modified to suit the needs of the database in which the SMTP Deux component has been installed.

SMTPd_Extract_EmailAddress

SMTPd_Extract_EmailAddress (*Formatted Recipient*) => *Email Address*

```

SMTPd_Extract_EmailAddress
(
    -> Formatted Recipient : Text
)
=> Email Address : Text

```

Parameter	Type	Description
<i>Formatted Recipient</i>	Text	Formatted recipient
<i>Email Address</i>	Text	Full name of recipient

The method *SMTPd_Extract_EmailAddress* will extract the email address within a properly formatted recipient. See the section **Basics of SMTP, Formatted Recipients** of this manual, above, for details on properly formatted recipients for use in the header of email being sent.

Formatted Recipient is the formatted recipient to extract a value from.

Email Address is email address extracted from *Formatted Recipient*.



SMTPd_Extract_FullName

SMTPd_Extract_FullName (*Formatted Recipient*) => *Full Name*

SMTPd_Extract_FullName

```
(
    -> Formatted Recipient : Text
)
=> Full Name : Text
```

Parameter	Type	Description
<i>Formatted Recipient</i>	Text	Formatted recipient
<i>Full Name</i>	Text	Full name of recipient

The method *SMTPd_Extract_FullName* will extract the full name within a properly formatted recipient. See the section **Basics of SMTP, Formatted Recipients** of this manual, above, for details on properly formatted recipients for use in the header of email being sent.

Formatted Recipient is the formatted recipient to extract a value from.

Full Name is full name extracted from *Formatted Recipient* .



SMTPd_Get_Attachment_w_Header

SMTPd_Get_Attachment_w_Header (*Referenced Attachment* ;
Attachment Filename ; *Content Type*) => *Error Code*

SMTPd_Get_Attachment_w_Header

```
(  
    -> Referenced Attachment : Pointer  
    -> Attachment Filename : Text  
    -> Content Type : Text  
)  
=> Error Code : Longint
```

Parameter	Type	Description
<i>Referenced Attachment</i>	Pointer	Referenced BLOB containing encoded document to include as document within formatted attachment
<i>Attachment Filename</i>	Text	Full filename of attachment formatted for sending through email
<i>Content Type</i>	Text	Content type of attachment being formatted for sending through email
<i>Error Code</i>	Longint	Error code returned from method for formatting attachment for sending through email

The method **SMTPd_Get_Attachment_w_Header** will format an encoded document for properly sending through email. This routine should be used in conjunction with the SMTP Deux method **SMTPd_Send_Message_Complex** only.

Referenced Attachment is a pointer to a BLOB containing the document to include as an attachment within the formatted attachment being created for sending through email. The contents of the referenced BLOB will be Base64 encoded (a standard encoding scheme for email attachments) for sending through email.

The resulting, formatted attachment will be the value referenced by *Referenced Attachment* after this method is successfully called.

Attachment Filename is the full document name of the attachment which is being formatted for sending through email.

Content Type is the type of content of the attachment being formatted for sending through email. Common values for content type include:

```
text/plain
text/html
image/gif
image/jpeg
application/pdf
application/octet-stream
application/x-macbinary
application/x-stuffit
```

See the section **Basics of SMTP, Content Types** in this manual, above, for more information on content types.

Error Code is the error code returned from calling this method for formatting an attachment for sending through email. The following tables lists all of the values returned from this method in *Error Code* :

<u>Value</u>	<u>Description</u>
0	no error, formatting done successfully
- 1	referenced attachment is empty
- 2	filename is empty
- 3	content type is empty



SMTPd_Get_Recipient_Formatted

SMTPd_Get_Recipient_Formatter (*Full Name* ; *Email Address*) =>
Formatted Recipient

SMTPd_Get_Recipient_Formatter

```
(
    -> Full Name : Text
    -> Email Address : Text
)
=> Formatted Recipient : Text
```

Parameter	Type	Description
<i>Full Name</i>	Text	Full name of recipient
<i>Email Address</i>	Text	Email address of recipient
<i>Formatted Recipient</i>	Text	Formatted recipient

The method *SMTPd_Get_Recipient_Formatted* will properly format a recipient line for inclusion in an email being sent. See the section **Basics of SMTP, Formatted Recipients** of this manual, above, for details on properly formatted recipients for use in the header of email being sent.

Full Name is full name of the recipient being formatted. It is valid to have an empty value for this parameter.

Email Address is the email address of the recipient being formatted

Formatted Recipient is the formatted recipient for the parameters specified in calling this method.



SMTPd_Send_Message_Attachment

SMTPd_Send_Message_Attachment (*From Formatted Address ; Referenced To Formatted Addresses ; Referenced CC Formatted Addresses ; Referenced BCC Formatted Addresses ; SMTP AUTH Enabled ; Username ; Password ; Subject ; Body ; SMTP Server Domain Name ; Reply-To Formatted Address ; Referenced Unencoded Attachment ; Attachment Filename ; Attachment Content Type*) => *Error Code*

SMTPd_Send_Message_Attachment

```
(
    -> From Formatted Address : Text
    -> Referenced To Formatted Addresses : Pointer
```

- > *Referenced CC Formatted Addresses* : **Pointer**
- > *Referenced BCC Formatted Addresses* : **Pointer**
- > *SMTP AUTH Enabled* : **Longint**
- > *Username* : **Text**
- > *Password* : **Text**
- > *Subject* : **Text**
- > *Body* : **Text**
- > *SMTP Server Domain Name* : **Text**
- > *Reply-To Formatted Address* : **Text**
- > *Referenced Unencoded Attachment* : **Pointer**
- > *Attachment Filename* : **Text**
- > *Attachment Content Type* : **Text**

)

=> *Error Code* : **Longint**

Parameter	Type	Description
<i>From Formatted Address</i>	Text	Formatted address to send email from
<i>Referenced To Formatted Addresses</i>	Pointer	Referenced text array containing formatted addresses for To: line
<i>Referenced CC Formatted Addresses</i>	Pointer	Referenced text array containing formatted addresses for CC: line
<i>Referenced BCC Formatted Addresses</i>	Pointer	Referenced text array containing formatted addresses for blind carbon copy recipients of email
<i>SMTP AUTH Enabled</i>	Longint	qi code for whether SMTP AUTH is to be used for authenticating SMTP access
<i>Username</i>	Text	Username to use for accessing SMTP server with SMTP AUTH
<i>Password</i>	Text	Password to use for accessing SMTP server with SMTP AUTH
<i>Subject</i>	Text	Subject of email
<i>Body</i>	Text	Contents of email
<i>SMTP Server Domain Name</i>	Text	Domain name of SMTP server to use for sending email
<i>Reply-To Formatted Address</i>	Text	Formatted address to include to email as a reply-to address
<i>Referenced Unencoded Attachment</i>	Pointer	Referenced BLOB containing unencoded document to include as attachment with email being sent
<i>Attachment Filename</i>	Text	Full filename of attachment being sent with email

<i>Attachment</i>	Text	Content type of attachment being sent with email
<i>Content Type</i>		
<i>Error Code</i>	Longint	Error code returned from method for sending email

The method *SMTPd_Send_Message_Attachment* will send a single email using the parameters provided. This method is the simplest and easiest means to send email with an attachment using SMTP Deux, providing basic parameters and options for the email to be sent.

From Formatted Address is the formatted address to use to send the email from. Due to limitations in the SMTP protocol, the formatted address listed as the From: need not be an actual email address; but , it is strongly considered to be "bad form" to send email with a From: line that is invalid.

Referenced To Formatted Addresses is a pointer to an array of formatted addresses to use in the To: line of the email and to include as recipients of the email. Pass in NULL for this parameter if no To: addresses are wanted.

Referenced CC Formatted Addresses is a pointer to an array of formatted addresses to use in the CC: line of the email and to include as recipients of the email. Pass in NULL for this parameter if no CC: addresses are wanted.

Referenced BCC Formatted Addresses is a pointer to an array of formatted addresses to include as recipients of the email. BCCed formatted addresses will not be listed in the header of the email as being recipients of any kind. Pass in NULL for this parameter if no BCC: addresses are wanted.

SMTP AUTH Enabled is a longint code for indicated whether SMTP AUTH is to be used to authenticate access to the SMTP server for sending email. Passing a value of one (1) will enable SMTP AUTH; passing a value of zero (0) will disable SMTP AUTH.

Username is the username to use when accessing the SMTP server using SMTP AUTH. If SMTP AUTH is not

enabled using the *SMTP AUTH Enabled* parameter, then the value of this parameter is ignored.

Password is the password to use when accessing the SMTP server using SMTP AUTH. If SMTP AUTH is not enabled using the *SMTP AUTH Enabled* parameter, then the value of this parameter is ignored.

Subject is the subject line of the email to be sent. There are no prohibitions against sending an email with an empty Subject: line, though it is considered to be "bad form" to do so.

Body is the full body of the email to be sent. There are no prohibitions against sending an email with no content, though it is considered to be "bad form" to do so. The contents of the body are restricted to common 4D restrictions for text variables, and should therefore be used only for textual bodies of 32000 bytes or less.

SMTP Server Domain Name is the full domain name of the SMTP server to use to send the email. This can be the IP address of the SMTP server, if needed; merely make certain in this case that the IP address is specified in a text format.

Reply-To Formatted Address is the formatted address to use in an optional Reply-To: line in the email to be sent. Pass in NULL for this parameter if no Reply-To: line is wanted. A Reply-To: line is optional in email and not all email clients support the Reply-To: line when replying to an email received.

Referenced Unencoded Attachment is a pointer to a BLOB containing the document to include as an attachment with the email to be sent. The contents of the referenced BLOB will be Base64 encoded (a standard encoding scheme for email attachments) for sending and included in the email in a standard format for reception. If no attachment is to be sent with the email then this parameter can be set to NULL.

Attachment Filename is the full document name of the attachment which is being sent with the email. If no attachment is being sent then this parameter can be set to NULL.

Attachment Content Type is the type of content of the attachment being sent with the email. Common values for content type include:

text/plain
text/html
image/gif
image/jpeg
application/pdf
application/octet-stream
application/x-macbinary
application/x-stuffit

If no attachment is being sent with the email then this parameter can be set to NULL.

See the section **Basics of SMTP, Content Types** in this manual, above, for more information on content types.

Error Code is the error code returned from calling this method for sending the email. If the sending of the email was successful, *Error Code* will be set to zero (0). Any problem encountered when trying to send the email will force *Error Code* to be set to a negative value.



SMTPd_Send_Message_Complex

SMTPd_Send_Message_Complex (*Referenced Mail Header IDs ; Referenced Mail Header Values ; SMTP Server Domain Name ; SMTP AUTH Enabled ; Username ; Password ; Referenced Body ; Content Type ; Referenced Array of Referenced Attachments*)
=> *Error Code*

SMTPd_Send_Message_Complex

(

- > *Referenced Mail Header IDs* : **Pointer**
- > *Referenced Mail Header Values* : **Pointer**

- > *SMTP Server Domain Name* : **Text**
- > *SMTP AUTH Enabled* : **Longint**
- > *Username* : **Text**
- > *Password* : **Text**
- > *Referenced Body* : **Pointer**
- > *Content Type* : **Text**
- > *Referenced Array of Referenced Attachments* : **Pointer**

)

=> *Error Code* : **Longint**

Parameter	Type	Description
<i>Referenced Mail Header IDs</i>	Pointer	Pointer to longint array containing mail header ID values corresponding to <i>Referenced Mail Header Values</i> , below
<i>Referenced Mail Header Values</i>	Pointer	Pointer to text array containing mail header values corresponding to <i>Referenced Mail Header IDs</i> , above
<i>SMTP Server Domain Name</i>	Text	Domain name of SMTP server to use for sending email
<i>SMTP AUTH Enabled</i>	Longint	qi code for whether SMTP AUTH is to be used for authenticating SMTP access
<i>Username</i>	Text	Username to use for accessing SMTP server with SMTP AUTH
<i>Password</i>	Text	Password to use for accessing SMTP server with SMTP AUTH
<i>Referenced Body</i>	Pointer	Pointer to BLOB containing body of email to be sent
<i>Content Type</i>	Text	Content type of body of email being sent
<i>Referenced Array of Referenced Attachments</i>	Pointer	Pointer to an array of pointers referenced formatted attachments to be sent with the email
<i>Error Code</i>	Longint	Error code returned from method for sending email

The method *SMTPd_Send_Message_Complex* will send an email with the parameter specified. This method is the most complex, and most flexible, means to send email using SMTP Deux, providing parameters and options for all areas of the email to be sent.

Referenced Mail Header IDs is a pointer to a longint array containing the email header IDs corresponding to

different email header lines. The list of accepted email header IDs are listed in the **Constants, SMTPd_Header_Codes** section of this manual, above. The array referenced by this parameter must have the same number of elements and correspond by index for values as that of the *Referenced Mail Header Values* parameter.

Referenced Mail Header Values is a pointer to a text array containing the email header values corresponding to different email header lines. The array referenced by this parameter must have the same number of elements and correspond by index for IDs as that of the *Referenced Mail ID Values* parameter.

All values in these arrays which exist for To:, CC:, and BCC: lines will be recipients of the email being sent. Of course, values in these arrays for BCC: recipients will not be listed in the actual headers of the email being sent. Values for each header line must be properly formatted. Specifically, the formatting of To:, CC:, BCC:, and Reply-To: values can be easily accomplished by using the SMTP Deux method entitled *SMTPd_Get_Recipient_Formatted*, documented above.

SMTP Server Domain Name is the full domain name of the SMTP server to use to send the email. This can be the IP address of the SMTP server, if needed; merely make certain in this case that the IP address is specified in a text format.

SMTP AUTH Enabled is a longint code for indicated whether SMTP AUTH is to be used to authenticate access to the SMTP server for sending email. Passing a value of one (1) will enable SMTP AUTH; passing a value of zero (0) will disable SMTP AUTH.

Username is the username to use when accessing the SMTP server using SMTP AUTH. If SMTP AUTH is not enabled using the *SMTP AUTH Enabled* parameter, then the value of this parameter is ignored.

Password is the password to use when accessing the SMTP server using SMTP AUTH. If SMTP AUTH is not enabled using the *SMTP AUTH Enabled* parameter, then the value of this parameter is ignored.

Referenced Body is a pointer to a BLOB containing the body of the email to be sent. The body of an email need not be just text, though it is commonly considered "bad form" to send an email with a body of any other content type. There are no restrictions at all, either for size or content type, for the body of the email to be sent when using this method. The contents of *Referenced Body* will be used directly as the content of the email to be sent.

Content Type is the type of content of the body of the email being sent. Common values for content type include:

- text/plain
- text/html
- image/gif
- image/jpeg
- application/pdf
- application/octet-stream
- application/x-machbinary
- application/x-stuffit
- multipart/mixed

If NULL is passed as the value for this parameter, the body content type will assume to be "text/plain" and the sent will reflect this assumption.

See the section **Basics of SMTP, Content Types** in this manual, above, for more information on content types.

Referenced Array of Referenced Attachments is a pointer to an array of pointers which referenced BLOBs containing the attachments to send with the email. Each attachment in the doubly referenced BLOBs must be encoded and formatted already for sending. Use the SMTP Deux method *SMTPd_Get_Attachment_w_Header*, documented above, to properly format attachments for sending in an email. Passing NULL for this parameter

will indicate that no attachments are to be sent with the email.

Error Code is the error code returned from calling this method for sending the email. If the sending of the email was successful, *Error Code* will be set to zero (0). Any problem encountered when trying to send the email will force *Error Code* to be set to a negative value.



SMTPd_Send_Message_Simple

SMTPd_Send_Message_Simple (*From Formatted Address ; To Formatted Address ; SMTP AUTH Enabled ; Username ; Password ; Subject ; Body ; SMTP Server Domain Name ; Reply-To Formatted Address*) => *Error Code*

SMTPd_Send_Message_Simple

```
(  
    -> From Formatted Address : Text  
    -> To Formatted Address : Text  
    -> SMTP AUTH Enabled : Longint  
    -> Username : Text  
    -> Password : Text  
    -> Subject : Text  
    -> Body : Text  
    -> SMTP Server Domain Name : Text  
    -> Reply-To Formatted Address : Text  
)  
=> Error Code : Longint
```

Parameter	Type	Description
<i>From Formatted Address</i>	Text	Formatted address to send email from
<i>To Formatted Address</i>	Text	Formatted address to send email to
<i>SMTP AUTH Enabled</i>	Longint	qi code for whether SMTP AUTH is to be used for authenticating SMTP access
<i>Username</i>	Text	Username to use for accessing SMTP server with SMTP AUTH
<i>Password</i>	Text	Password to use for accessing SMTP server with SMTP AUTH
<i>Subject</i>	Text	Subject of email
<i>Body</i>	Text	Contents of email

<i>SMTP Server Domain Name</i>	Text	Domain name of SMTP server to use for sending email
<i>Reply-To Formatted Address</i>	Text	Formatted address to include to email as a reply-to address
<i>Error Code</i>	Longint	Error code returned from method for sending email

The method *SMTPd_Send_Message_Simple* will send a single email using the parameters provided. This method is the simplest and easiest means to send email using SMTP Deux, providing basic parameters and options for the email to be sent.

From Formatted Address is the formatted address to use to send the email from. Due to limitations in the SMTP protocol, the formatted address listed as the From: need not be an actual email address; but , it is strongly considered to be "bad form" to send email with a From: line that is invalid.

To Formatted Address is the formatted address to send the email to and to include in the To: line as a recipient of the email.

SMTP AUTH Enabled is a longint code for indicated whether SMTP AUTH is to be used to authenticate access to the SMTP server for sending email. Passing a value of one (1) will enable SMTP AUTH; passing a value of zero (0) will disable SMTP AUTH.

Username is the username to use when accessing the SMTP server using SMTP AUTH. If SMTP AUTH is not enabled using the *SMTP AUTH Enabled* parameter, then the value of this parameter is ignored.

Password is the password to use when accessing the SMTP server using SMTP AUTH. If SMTP AUTH is not enabled using the *SMTP AUTH Enabled* parameter, then the value of this parameter is ignored.

Subject is the subject line of the email to be sent. There are no prohibitions against sending an email with

an empty Subject: line, though it is considered to be "bad form" to do so.

Body is the full body of the email to be sent. There are no prohibitions against sending an email with no content, though it is considered to be "bad form" to do so. The contents of the body are restricted to common 4D restrictions for text variables, and should therefore be used only for textual bodies of 32000 bytes or less.

SMTP Server Domain Name is the full domain name of the SMTP server to use to send the email. This can be the IP address of the SMTP server, if needed; merely make certain in this case that the IP address is specified in a text format.

Reply-To Formatted Address is the formatted address to use in an optional Reply-To: line in the email to be sent. Pass in NULL for this parameter if no Reply-To: line is wanted. A Reply-To: line is optional in email and not all email clients support the Reply-To: line when replying to an email received.

Error Code is the error code returned from calling this method for sending the email. If the sending of the email was successful, *Error Code* will be set to zero (0). Any problem encountered when trying to send the email will force *Error Code* to be set to a negative value.

Version History

The following is a brief version history of the SMTP Deux component. It details release notes, bug fixes, and changes for each version publicly available.

SMTP Deux v1.0.0

released 20010906

Changes:

Updated the component to require TCP Deux v1.0.1 or above.

Added the methods `ENV_Get_SMTPd_HardName_Long` and `ENV_Get_SMTPd_HardName_Short`.

SMTP Deux v1.0.0b03

released 20010707

Changes:

Add support for SMTP AUTH throughout all email sending routines. SMTP AUTH support consists of dynamic support for CRAM MD5, Login, and Plain.

Modified the methods **SMTPd_Send_Message_Attachment**, **SMTPd_Send_Message_Complex**, and **SMTPd_Send_Message_Simple** to now accept three additional parameters. The three parameters are used to indicate whether SMTP AUTH is to be used, and the user and password to use for SMTP AUTH.

Add a TRACE call to the **SMTPd_ERROR** method.

SMTP Deux v1.0.0b02

released 20010629

Changes:

Removed comment stating to only use in conjunction with send message complex in routine

SMTPd_Get_Recipient_Formatted.

Created two new routines **SMTPd_Extract_FullName**, **SMTPd_Extract_EmailAddress**; these will extract data from a formatted SMTP recipient.

Modified the routine **SMTPd_Send_Message_Simple** so that the From:, To:, and Reply-To: addresses have to be sent to it formatted with **SMTPd_Get_Recipient_Formatted** beforehand.

Modified routine **SMTPd_Send_MAIL_Command** so it extracts the From: address from the fully formatted From: parameter before sending MAIL command; also wrapped the unformatted From: address with "<>".

Modified routine **SMTPd_Send_RCPT_Command** so it extracts the recipients address from the formatted addresses, checks to make sure that it was received, and then proceeds to build message to send.

Modified the routine **SMTPd_MH_Add_Recipients** so that it builds the recipients for the mail header without the full name parameter, since the full name should already be formatted with the address.

Modified the routine **SMTPd_Send_Message_Attachment** so that all of the addresses have to be formatted with Modified before hand.

Modified routine **SMTPd_Send_Message_Complex** to build address list with them still formatted; also checks the addresses with the new routine **SMTPd_Extract_Address** for proper formatting.

SMTP Deux v1.0.0b01

released 20010518

Changes:

First public release of the component.

SMTP Deux Error Codes

asd

Index

asd